ADAPTIVELY WEIGHTED, PARTITIONED CONTEXT EDIT DISTANCE STRING MATCHING

TECHNICAL FIELD OF THE INVENTION

This application is a continuation-in-part of U.S. patent Application, Serial No. 09/294,701, filed April 19, 1999 entitled: "Method and System for Generating Structured Data From Semi-Structured Data Sources", and is incorporated herein by reference in its entirety.

The invention relates generally to pattern matching and more particularly to the partitioning of a pattern to be matched into components which are individually weighted and individually compared to a test string to determine corresponding edit distances which are used to arrive at a composite score for candidate matches of the pattern.

BACKGROUND OF THE INVENTION:

The increasing use of means such as the Internet to distribute information presents potential users of that information with a daunting task - to find and extract desired information from various sources. For example, numerous web sites on the Internet maintain databases of information which are made freely available to the public. Users need only request the information, and it will be served to them. The users who request this information, however, may get more information than they desired. The specific information which is actually desired by a particular user may be buried in an enormous amount of information. Consequently, the task of identifying and extracting the desired information may be overwhelming. It may therefore be useful to have a tool for recognizing and extracting the desired information.

Tools of this sort currently employ string matching technology which is based upon either regular expressions or inductive learning. These technologies, however suffer from some significant disadvantages. For example, regular expressions can only be used when there is a well-defined pattern for the value to be matched. Systems which employ regular expressions for string matching also require substantial training which typically involves tedious (and often error-prone) programming using detailed, domain-specific knowledge of the pattern which is to be matched.

Even if the desired pattern can be well-defined in a system that uses regular expressions, only exact matches will be identified. Even minor differences between the pattern and the string that is being searched (which may be unimportant to the person who desires information) may prevent a potential match to be ignored. For example, differences in font tags (e.g., font, color, emphasis) or value formats, or the

substitution, insertion or deletion of text, or even single characters can cause these systems to fail to identify matches that would be acceptable to the user. Thus, if such minor changes are made, regular-expressions must be redefined in order to identify these matches.

Systems that employ inductive learning to perform pattern matching share some of these drawbacks. These systems require many examples of potential matches in order to train them. These examples must span the population from which the matches are to be extracted. Consequently, a substantial amount of time and resources are required to adequately train these systems. Even when they are adequately trained, inductive learning systems achieve a relatively low accuracy rate (on the order of 70%.)

SUMMARY OF THE INVENTION:

One or more of the problems outlined above may be solved by the various embodiments of the invention. Broadly speaking, the invention comprises a system and method for examining a string of symbols and identifying portions of the string which match a predetermined pattern which is partitioned into a prefix, a value and a suffix. In one embodiment, the prefix, value and suffix components of the pattern are adaptively weighted, and corresponding edit distances between the respective components and candidate matches in the string are calculated. The context components of the pattern (the prefix and suffix) may be weighted so that the pattern matches are based on the context of the attribute which is being searched rather than the value component of the pattern. Keywords or regular expressions may be used to filter the candidate value matches for the pattern.

In one embodiment, the present system and method are implemented using dynamic programming techniques. These techniques allow for efficient computation in this arena. A single example may be used to provide a pattern for searching a target string. The pattern of the example is partitioned into a prefix, a value and a suffix. An edit distance matrix is constructed using dynamic programming for efficient computation, for the prefix to identify candidate matches for the prefix. The prefix edit distance matrix includes a row of indicator cells that define the ends of one or more prefix candidates and the beginnings of a one or more corresponding value windows. An edit distance matrix is then constructed for the suffix to identify candidate matches for the suffix. The suffix edit distance matrix includes a row of alignment cells that define the beginning of one or more suffix candidates and the ends of one or more corresponding value windows. One or more value candidates are defined by the

value windows. The cost of each contiguous prefix-value-suffix combination (each candidate match for the entire pattern) is determined and one or more of the best candidates are selected. This selection may, for example, be based on the cost of the corresponding candidates falling below a certain threshold, it may be based on the candidates' cost being within a certain tolerance of the lowest cost, or it may be based on a predetermined number of lowest cost candidates.

BRIEF DESCRIPTION OF THE DRAWINGS:

Other objects and advantages of the invention may become apparent upon reading the following detailed description and upon reference to the accompanying drawings in which:

FIGURE 1 is a diagram illustrating the partitioning of a pattern (r) into prefix (p), value (v) and suffix (s) components.

FIGURE 2 is a diagram illustrating the partitioning of a target string (t) into components which correspond to the prefix (p), value (v) and suffix (s) components of a pattern.

FIGURE 3 is a diagram illustrating the matching of the components of a pattern (r) to the components of a target string (t).

FIGURES 4a-4f are flow diagrams illustrating the manner in which a pattern is partitioned and candidate matches for the pattern are identified in one embodiment.

FIGURE 5 is an example of the edit distance matrices for a particular pattern and target string in one embodiment.

FIGURE 6 is a diagram illustrating the manner in which the edit distance value for a particular cell is determined based on the values of surrounding cells in one embodiment.

FIGURE 7 is a diagram illustrating the convergence of the lowest cost paths corresponding to a range of indicator cells in one embodiment.

While the invention is subject to various modifications and alternative forms, specific embodiments thereof are shown by way of example in the drawings and the accompanying detailed description. It should be understood, however, that the drawings and detailed description are not intended to

limit the invention to the particular embodiment which is described. This disclosure is instead intended to cover all modifications, equivalents and alternatives falling within the spirit and scope of the present invention as defined by the appended claims.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT:

An exemplary embodiment of the invention is described below. It should be noted that this and any other embodiments described below are exemplary and are intended to be illustrative of the invention rather than limiting.

One embodiment of the present invention comprises a method for examining a string of symbols and identifying portions of the string which match a predetermined pattern using adaptively weighted, partitioned context edit distances. The method involves partitioning the pattern into components (e.g., context components and a value component) and identifying candidate matches for each of the components. The candidate matches for each component are determined by calculating an edit distance between that component and each potentially matching set (sub-string) of symbols within the string. The edit distances corresponding to the components may be weighted.

In one embodiment, the components of the pattern are weighted so that the pattern matches are based on the context of the attribute which is being searched rather than simply on the attribute's own pattern. This may be especially important when searching for attributes that may not have a predetermined pattern. The present system and method require minimal training (e.g., a single example) and the training requires no programming or detailed knowledge of the pattern of the searched value. The use of edit distance calculations as the basis for scoring candidate matches allows the matching algorithm to degrade gracefully with changes in the target string. These features result in a system and method which are highly accurate in the identification of matches.

The present system and method are contemplated to be useful for identifying patterns within various types of

dynamic, unstructured or semi-structured data. One example of this type of data would be web pages of the type that are retrievable via the internet. Web pages may contain data that varies from one source to another and may themselves change with time. Thus, while data of interest (i.e., acceptable matches to a particular pattern) may remain available to a user, it may change form (e.g., it may be presented in different colors or fonts.) The present system and method provide a means to identify pattern matches despite these changes.

The present system and method are used to identify particular patterns within a string of symbols. A string, as used herein, is simply a finite sequence of symbols. These symbols may belong to an arbitrary group of symbols. This group of symbols may be referred to as an alphabet. For the purposes of this disclosure, the pattern string will generally be identified as *r*, while the string which is searched to identify matches for this pattern will generally be identified as *t*. The string to be searched may also be referred to as the input, test, or target string. The particular sub-strings of the target string that are compared to the pattern (or components of the pattern) may be referred to as potential matches or candidate matches.

The edit distance between a pattern and a potential match for the pattern is simply a measure of the differences between them. The edit distance may be defined in a variety of different ways in different embodiments. For example, in one embodiment, if the alphabet used is the characters contained on a typewriter, the edit distance may be the minimal number of keystrokes on a typewriter which would be required to change a candidate string to an identical match of the pattern string.

While it should be remembered that the functions by which edit distance is calculated may be arbitrarily defined, the analogy of edit distance to keystrokes of a typewriter will be used for the purposes of this disclosure. Accordingly, three edit operations will be defined. The *insert* operation inserts a symbol *a* into the string *t* at position *i*. The *delete* operation deletes a symbol *a* from the string *t* at position *i*. The *replace* operation replaces a symbol *a* at position *i* in the string *t* with a symbol *b*. (In some instances, it may also be useful to define an operation *match*, in which a symbol *a* at position *i* in the string *t* is replaced with a symbol *a*.)

Each of the edit operations (insert, delete, replace) has a cost *c* associated with it. The cost functions associated with the insert, delete and replace operations are C_i, C_d and C_r, respectively. The cost of a particular edit operation e_1 is $c(e_1)$. The cost of a sequence of edit operations is cumulative. Consequently, if a sequence of edit operations *L* comprises *n* edit operations, the total cost of the sequence of operations $C(L)$ is equal to $c(e_1) + c(e_2) + \ldots + c(e_n)$. For the purposes of the examples contained in this disclosure, the cost functions C_i, C_d and C_r will be assumed to be 1 unit per keystroke and the cost of a match is 0 units. These cost functions may be defined differently in the various embodiments of the system and method.

For string *r* and a potentially matching string ${}_it_j$ (where *r* is the pattern and ${}_it_j$ is the string to be matched) the edit distance can be defined as the minimal edit distance which can be achieved for the potentially matching string ${}_it_j$. It should be noted that the edit distance is defined to be the minimal edit distance because any potentially matching string can be edited in a number of ways to produce a match to a particular pattern. The edit distance for the potentially matching string should therefore be defined in this manner. The edit

distance between strings r and ${}_it_j$ can also be expressed as $ED(r, {}_it_j) = \min \{C(L)\}$ where L exists within $\boldsymbol{L}$ and where $\boldsymbol{L}$ is a collection of all finite sequences of edit operations which transform ${}_it_j$ into r.

As indicated above, the present system and method do not simply attempt to match the entire pattern r, but instead partition the pattern into components which can be individually matched against the test string t. This is illustrated in FIGURE 1. In this figure, string r is illustrated simply as a line. The line represents the string of symbols which comprise r. r can be divided into a prefix string p, a value string v and a suffix string s. The prefix and suffix component strings can be referred to as the context of component string v. In the case of a pattern extracted from a web page, HTML tags surrounding a word may serve as the context components of the search pattern.

A string of symbols ${}_it_j$ can be broken into prefix, value and suffix components in the same manner as pattern r. This is illustrated in FIGURE 2, where the string t is represented by the horizontal line. A potentially matching string ${}_it_j$ includes symbols i through j. The prefix component (p) comprises symbols i through k, the value component (v) comprises symbols $k+1$ through l, and the suffix component (s) comprises symbols $l+1$ through j, where $i \leq k \leq l \leq j$. String components p, v and s can also be identified as ${}_it_k$, ${}_{k+1}t_l$ and ${}_{l+1}t_j$, respectively.

If the potentially matching string ${}_it_j$ can be edited to match pattern r, then the prefix, value and suffix components of ${}_it_j$ can each be edited to match the corresponding prefix, value and suffix components of r. This is illustrated in FIGURE 3. Consequently, an edit distance can be calculated between the pattern prefix and potentially matching prefix,

between the pattern value and potentially matching value, and between the pattern suffix and potentially matching suffix. The edit distance between the pattern r and the string ${}_it_j$ for a given k and l can be estimated by the sum of these component edit distances. Then, since the edit distance between the pattern r and the string ${}_it_j$ is the minimum cost of the possible edit sequences, it can be defined as $ED(r, {}_it_j) = \min \{ED(p, {}_it_k) + ED(v, {}_{k+1}t_l) + ED(s, {}_{l+1}t_{j,})\}$ over all $_k$ and $_l$ with $i \le k \le l \le j$. It should be noted that the edit distance is the minimum of the sum of the prefix, value and suffix (which are consecutive and contiguous) rather than the sum of the minima of the prefix, value and suffix.

This is useful in searching dynamic or unstructured data such as web pages because regular expressions break down easily when dealing with this type of data. For instance, in the case of web pages, the delimiters included on either side of the value pattern in the underlying HTML comprise the context of the value. If the value is a product name, it may be surrounded by HTML tags which indicate that it is a product name. By searching for a value that is similar to this value, but not necessarily an exact match (e.g., the product name is different than the training example) in the same context as the search pattern, (e.g., it is surrounded by product name HTML tags) matches can be retrieved that may be intuitive, but may not satisfy a regular expression of the type used in the prior art. This may be achieved by weighting the context of the pattern much more heavily than the value in the scoring. Thus, if the pattern value is a single word, a match may consist of an entire paragraph in the same context. On the other hand, a word which is identical to the value, but exists in a different context may not be selected as a match.

The present method is also very useful for identifying matches in strings that are dynamic in nature. Referring

again to the example of web pages, web site operators often change their web pages in ways that do not substantially affect the substance of the desired information, but may cause a regular expression to fail. For instance, text may be reformatted to use a different font (e.g., italics.) If a regular expression is used to search for patterns that include tags for a first font, the expression will likely fail if new or different font tags are used. Using the present method, the edit distances corresponding to candidate matches will change, but they will do so gracefully. That is, the scores of the candidate matches may change, but the best matches will likely remain the best matches and will therefore still be selected despite minor changes to the searched data.

Referring to FIGURES 4a-4f, a flow diagram illustrating one embodiment of the present method is shown. The method will be described in more detail below in relation to the figures. Generally speaking, this method comprises identifying candidate matches for a pattern within interesting spans of the target string, scoring candidate matches based upon their respective context edit distances, and selecting one or more matches based on these scores. Effectively, the pattern is compared to all possible segments of the target string and the highest scoring (lowest cost) matches are identified. It should be noted that "scoring" as used herein refers generally to assigning an edit distance or other cost value to a potential match. A "better" score is typically a lower cost or edit distance.

FIGURE 4a depicts a first portion of the method in which a weighting function is determined for the respective components of pattern *r* (i.e., prefix *p*, value *v* and suffix *s*.) the weighting function is calculated based upon keyword selection, value format and other factors. A threshold value is then set based on the weighting function and pattern *r*.

The threshold is set to a percentage of the worst possible edit distance, where the percentage can be selected by the user. This threshold may be adjusted in post-processing, if necessary to obtain a desired number of matches. After the weighting function and threshold values have been determined, it is determined whether pattern *r* has a special value format. If the pattern has a special value format, only strings which contain values that meet this format can be considered as matches to the pattern. Strings for which the values do not conform to the special format are discarded. If it is determined that pattern *r* has a special value format, the method branches to point B. Otherwise, it branches to point C.

Referring to FIGURE 4b, a portion of the method in which special value format patterns are matched is shown. Beginning at point B, a mapping from the string being searched, *t*, to the special value format is created. This mapping may, for example, effectively enable regular expressions that enforce the special value format to operate on displayable text without being affected by HTML tags. Using this mapping, those values within the string that conform to the special value format are identified as candidate values. The candidate values are then mapped back to the target string, *t*. Then, for a span surrounding each candidate value, context edit distances are determined for the potential prefix and suffix matches. The weighting function is then applied to generate a score for the candidate match corresponding to the candidate value. A filtering function may be applied if desired, for example to consolidate overlapping matches. If the match passes the filter, it is added to a match list. The threshold value T may be applied at this point, or applied during post-processing of the candidate matches' scores.

Referring to FIGURE 4c, a portion of the method involving the processing of patterns that are not constrained to conform to a special value format is shown. Beginning at point C, it is first determined whether the pattern *r* includes a keyword. If there is a keyword in the pattern, it must be matched exactly. If the keyword is not matched in a particular string, the string is effectively discarded as a match by ensuring that the score of the string is sufficiently high (e.g., infinite) that it will never be considered a match. If it is determined that pattern *r* includes a keyword, target string *t* is scanned to find all matches of the keyword. For each of the identified keywords in the string, a span of the string is constructed. Each span, because it surrounds a keyword, represents a region of interest within the string. If, on the other hand, pattern *r* does not contain a keyword, a single span comprising the full length of target string *t* is constructed. After the appropriate spans within target string *t*are identified, the method proceeds to point D, where the spans are processed to identify potential matches of pattern *r* based upon context edit distances within the spans.

It should be noted that a keyword for a pattern may be contained within the prefix, value or suffix, or it may be external to, but associated with the pattern. The keyword may also comprise a set of keywords, only one of which has to be matched. More generically, the keyword may comprise any condition or set of conditions which must obtain in order for a set of symbols to be considered a match for the pattern.

Referring to FIGURE 4d, a portion of the method involving the calculation of context edit distances and the identification of indicators and alignments of prefixes and suffixes is shown. Beginning at point D, a prefix edit distance matrix is constructed for the identification of potential matches of prefix *p* within a particular span. This

edit distance matrix covers the length of the span and contains values that represent the edit distance calculation between *p* and respective groups of symbols within *t*, as weighted by the weighting function indicated in FIGURE 4a. The edit distance matrix contains a row of indicator cells. Cells that have respective edit distances which fall below the threshold T are indicative of potential matches to prefix *p*. These below-threshold cells form an indicator set that may be filtered using information about the expected value type. The filtered indicators are then used as the basis for creating a set of non-overlapping sub-spans within the original span. These spans, characterized by low-weighted edit distances corresponding to prefix *p*, represent regions of interest within the original span. Further edit distance matrices (suffix matrices) are constructed for each of these identified regions of interest. These suffix edit distance matrices are based on the suffix of pattern *r* rather than the prefix. The edit distances are again calculated using weighting function indicated in FIGURE 4a. For each of the suffix edit distance matrices, a set of indicator cells that have edit distances below a predetermined threshold are identified. These indicator cells are traced backwards through the suffix edit matrices to find corresponding alignment positions. The best-scoring indicator corresponding to each of the alignment positions is retained.

Referring to FIGURE 4e, a portion of the method involving the identification of candidate value windows is shown. As shown in this figure, each of the potential value windows (i.e., each potential indicator-alignment pair) is examined to determine the lowest scoring candidate match. Beginning at point E, one of the indicators identified from the prefix edit distance matrix is selected. Based upon the position of the indicator and the corresponding alignments which were

identified from the suffix edit distance matrices, a maximum extent of the potential value corresponding to the indicator is determined. Then, each of the potential value windows between the indicator and the corresponding alignments is individually examined. First, it is determined whether the value window contains a valid value. If so, a score is calculated for the value window and it is determined whether this score falls below a predetermined threshold. If the score is below the threshold, it is determined whether the score is the best score of the value windows that have been examined so far. If the value window has the best score, it is designated as the best match. Thus, the best match for each indicator is identified. The best match for each of the indicators is then added to a match list that identifies the candidate matches for the pattern.

Referring to figure 4f, a portion of the method involving the final determination of the best match for the target pattern is shown. As illustrated in this figure, each candidate match is assigned a score based on the edit distance of the prefix, suffix and value. Then, based upon these scores, one or more of the best matches for the pattern are selected.

Referring to FIGURE 5, an example of the edit distance matrices described in connection with FIGURES 4a-4f is shown. This figure includes three matrices corresponding to a pattern *r* which includes a prefix, a value and a suffix. The first matrix is used to identify candidate matches for the prefix. The third matrix is used to identify candidate matches for the suffix. Finally, the second matrix is used to identify value windows based upon indicators from the first matrix and alignment positions from the third matrix.

In the example of FIGURE 5, pattern *r* comprises the symbol string "<TD> Price : $ 5 . 99 <TD> SKU : 345671". This string can be parsed into prefix "<TD> Price :", value "$ 5 . 99" and suffix "<TD> SKU : 345671". The prefix, value and suffix can further be parsed into individual symbols. It should be noted that, in some embodiments each letter or character may be considered a symbol. In the example illustrated in FIGURE 5, however, the symbols in the alphabet comprise groups of characters. For example, the prefix includes three symbols ("<TD>", "Price" and ":",) some of which contain multiple characters.

In the example illustrated in FIGURE 5, test string *t* comprises the symbol string "<TD> Description: Deluxe Snow Shovel <TD> Price : $ 7 . 99 <TD> SKU : 45893 <TD> SPECIAL </TD>". Again, in this embodiment, the symbols may each comprise multiple, rather than single, characters.

Each of the matrices in FIGURE 5 includes a column corresponding to each of the symbols in test string t and a row corresponding to each of the symbols in pattern r. The symbols of the test string are shown at the top of the prefix matrix above the corresponding columns. Likewise, the symbols of the pattern are shown at the left of each of the corresponding rows of the matrices, wherein the prefix symbols are to the left of the prefix matrix, the value symbols are to the left of the value matrix and the suffix symbols are to the left of the suffix matrix.

The edit distance matrices contain values representing the edit distances between patterns and test string segments corresponding to the rows and columns of the matrices. Put another way, each matrix cell has a corresponding row and column. The row of the cell represents a particular pattern and the column of the cell represents a particular test

string. The cell contains the accumulated edit distance between the particular pattern and the particular test string.

The particular pattern and test string (or segments thereof) which correspond to a cell are to some extent cumulative. For instance, the first row of a matrix corresponds to the first symbol of a pattern. The second row corresponds to the string comprising the first and second symbols of the pattern. The third row corresponds to the string comprising the first, second and third symbols of the pattern, and so on. The columns of the matrix can similarly be representative of the cumulative test string.

Each matrix is generally filled in from left to right and from top to bottom. The value for a cell can be determined in relation to one of three cells: the cell above it; the cell to the left of it; and the cell to the upper left of it. Put another way, the cell is one edit operation away from each of these adjacent cells and, upon completion of the appropriate edit operation, an exact match will result. For example, referring to FIGURE 6, a match in cell 20 can be achieved by starting at cell 21, consuming a symbol from the input (test) string, and adding the cost of the insert operation to the cost associated with cell 21. A match in cell 20 can also be obtained by starting at cell 22, consuming a symbol from the pattern string, and adding the cost of the delete operation to the cost associated with cell 22. Finally, a match in cell 20 can be obtained by starting at cell 23, consuming a symbol from the pattern string and a symbol from the input string, and adding the cost of the replace/match operation to the cost associated with cell 23. The value that is entered in the cell is the minimum cost required to reach that cell from one of these three cells. Further, the value entered in the cell

is the minimum accrued edit cost required to achieve a match of the corresponding pattern and test strings.

Any particular cell in a matrix can be filled with a corresponding edit cost when the cells above, to the left and to the upper left are filled. The order in which they are filled is arbitrary. In one embodiment, the cells of the top row are filled from left to right, then the cells of the next lower row are filled from left to right, and so on until the matrix is completely filled. When the matrix has been filled, each cell in the bottom row of cells contains the sum of the cells traversed to reach that cell. Thus, values in the bottom row of the matrix represent the cost for the entire pattern to be matched to a segment of the test string.

Because the cells in the bottom row of the edit distance matrix indicate the lowest cost (edit distance) to match the pattern to a corresponding test string segment, these cells are referred to herein as indicator cells. The indicator cells correspond to the last symbol in the candidate match (the test string segment.) If the last symbol in the candidate match is known, the first symbol in the candidate match can be found by following the path of lowest cost backward from the indicator cell to the top row of the matrix. This may be accomplished by using a preferred canonical order to resolve ties. As described above, the value contained in each cell is determined by taking the value in one of the surrounding (top, left or upper left) cells and adding the cost of performing the edit operation that would be required to move to the current cell. Thus, each cell effectively defines the last step in a path of minimal cost to reach that cell. The cell at the top of the matrix from which this path originates is referred to herein as an alignment cell or alignment position.

Referring again to the example of FIGURE 5, three edit distance matrices are shown. Each of the matrices has columns corresponding to the symbols of test string *t*. The rows of each matrix correspond to the symbols of the respective portion of pattern *r* (i.e., the prefix, the value and the suffix.) Each of the three matrices is essentially handled independently of the other two, in that they are each used to independently identify particular cells that identify a value window and determine the edit distance of the value in that window as compared to the value in the pattern. This is explained in more detail below.

The first (prefix) matrix is used to identify one or more indicator cells that define the beginning of a value window. The matrix is filled in as described above, resulting in the assignment of values to each of the indicator cells in the bottom row of the matrix. These cells are then examined to identify the lowest values, which correspond to the best candidate matches to the prefix of the pattern. In FIGURE 5, cell A as a value of 0, indicating that the corresponding segment of the test string is an exact match to the prefix of the pattern. The lowest cost path for cell A is therefore a 0-cost path which is indicated by the arrow extending from cell A through the cells to its upper left. The arrow terminates at the column corresponding to the first symbol in the match. (It should be noted that, if there were no exact match for the prefix, then the next-best candidate matches for the prefix might be indicated by the values of 1 in cells B and C.)

After the prefix matrix is filled in and indicator cells are identified, the suffix matrix is processed. This is a consequence of the desire to perform context-based matching. In other words, it is important to find the desired context (prefix and suffix) for the match, from which a value window

(the value match) can be identified. The value matrix is filled in after the prefix and suffix matrices so that the cost associated with particular value matches can be determined. (It should be noted that, in one embodiment, the value match is given very little weighting, so any candidate value match will be acceptable, as long as any value format requirements are met.)

The suffix matrix is filled in the same way the information for the prefix matrix is entered. In one embodiment, the values for the cells are entered from left to right in a first row, then from left to right in each succeeding row. When the matrix has been completely filled in (or at least within a certain span,) the indicators in the bottom row are used to identify the best candidate matches for the suffix of the pattern within the test string. The lowest cost path for these indicators is then followed back to the top of the matrix to identify candidate alignment positions. These alignment positions define the possible trailing ends of the value windows corresponding to the prefix matrix indicators (which define the starting ends of the windows).

In the example illustrated in FIGURE 5, it should be noted that there are no identical matches for the suffix of the pattern. Consequently, there are no indicator cells that have values of 0. The indicator cells instead have values that range from 1 to 4. In one embodiment, the indicators having the lowest scores are used as the basis for determining potential alignment positions. The lowest cost path for each of the lowest scoring indicator cells is tracked to the top of the matrix to identify the corresponding alignment positions. In FIGURE 5, the indicator cells having the lowest values are cells D and E. The lowest cost paths for both of these indicator cells lead to the same alignment position - cell F (which corresponds to symbol "<TD>" in both the pattern and

input). It should be noted that the lowest-cost paths for each of cells D and E converge to a single path after traversing one cell. The convergence of the paths is typical for nearby indicator cells. This is illustrated more clearly in the example below.

In one embodiment, rather than selecting only the lowest cost indicator cells of the suffix matrix, cells having a range of values are selected. Depending upon the distance between the indicator cells and the depth of the matrix (i.e., the length of the suffix pattern string) the lowest cost paths for all of these indicator cells will tend to converge. Referring to FIGURE 7, an example of the convergence of the lowest cost paths corresponding to a range of indicator cells is shown. In this figure, the suffix matrix of FIGURE 5 is repeated. Rather than selecting only indicators having values of 1, however, indicator cells having values of 2 or less are selected. These cells comprise two groups. The group on the left converges to the same symbol ("<TD>") as the best match for the prefix end, so the suffix match can be discarded (since no value match is possible.) The group of indicators on the right all converge to cell F. Using cell F as the best-match alignment position, the candidate value window extends from "$" to "99".

The edit distance matrices can be implemented for an entire target string, or only for selected spans of interest within the target string. The same process is applied in each of the selected spans, typically resulting in the selection of a best match for the pattern in each span. In some embodiments, however, more than one candidate match may be selected. These candidate matches can be filtered as they are identified, or they may be collected and filtered in post-processing after all the candidate matches have been identified. The filtering may, for example, comprise using a

threshold cost or score to identify all matches which are above or below a certain threshold, the n best matches, or all matches within a certain tolerance of the best scoring match.

The present method can be implemented in a variety of computer systems. In one embodiment, a general purpose computer which is coupled to the internet and configured to receive web pages may be configured to retrieve certain information (the pattern) from particular web pages (the target string.) The computer can be trained by retrieving a web page that contains an exemplary symbol string (HTML data.) Certain items within the web page may be selected by a user to define the pattern. The computer is configured to partition this pattern into components and proceed with the identification of candidate matches as described above. In yet another embodiment, the method may be embodied in the instructions stored on a computer-readable medium. This medium may be used on a general purpose computer to perform the method as described above.

While the present invention has been described with reference to particular embodiments, it should be understood that the embodiments are illustrative and that the scope of the invention is not limited to these embodiments. Many variations, modifications, additions and improvements to the embodiments described above are possible. It is contemplated that these variations, modifications, additions and improvements fall within the scope of the invention as detailed within the following claims.